Research Internship
Data Science

Radboud University

Multitask Wav2Vec2: Speech Recognition, Speaker-Change Detection & Speaker Recognition

Author:
BSc. Tijn Berns

First supervisor:
Prof. Dr. Ir. David van Leeuwen

Second supervisor:
MSc. Nik Vaessen

First assessor:
Prof. Dr. Ir. David van Leeuwen

Second assessor:
Dr. Louis ten Bosch

June 16, 2023

ABSTRACT

This research explores the diversity of the Wav2Vec2 network by fine-tuning it to perform three distinctive speech related tasks. The tasks we consider are automatic speech recognition (ASR), speaker-change detection, and speaker recognition. Our approach relies on the introduction of speaker-change tokens or speaker-identity tokens to the target transcriptions during network fine-tuning. In this work, we introduce a method for extracting speaker embeddings from the Wav2Vec2 network, and show that our approach allows the model to perform the three distinctive tasks on an artificially created multi-utterance dataset. Additionally, we show that the introduction of the tokens during network fine-tuning is beneficial for the ASR performance.

1 INTRODUCTION

In recent years, there has been a tendency for larger neural models to be trained on large amounts of labeled data. An important bottleneck here being computational power and the amounts of labeled data [7]. This trend is shifting to a training framework where the models are pre-trained on large amounts of unlabeled data and fine-tuned on limited amounts of labeled data. In various natural language processing (NLP) tasks, this framework has shown great successes [9, 21, 27].

The introduction of Wav2Vec2 [5] showed that the framework of pre-training and fine-tuning can also be successfully applied in automatic speech recognition (ASR). Recent research has shown that the network architecture with its training strategy does not only obtain state-of-the-art ASR performance, but can also be successfully applied in other speech related tasks such as speaker recognition [23] and language identification [10].

In this research, we further explore the diversity of Wav2Vec by fine-tuning it to perform multiple tasks: speech recognition, speaker recognition, and speaker-change-detection. Our approach to such a multitask system relies on the introduction of speaker identity or speaker-change tokens to target transcription during network fine-tuning. The research makes the following contributions:

- We show that a Wav2Vec2-Base network fine-tuned using transcriptions containing speaker identity tokens is able to transcribe multi-utterance recordings and indicate the moments at which speakers change in the transcribed text.
- We introduce a method for extracting speaker embeddings at time steps at which the speaker in the recording changes.
- We show that by introducing speaker-change symbols or speaker identities to the transcriptions of multi-participant recordings, we can improve ASR performance.

2 BACKGROUND

In this section, we first briefly cover the most relevant work related to each task we consider for our multitask Wav2Vec2 network: automatic speech recognition, speaker-change detection, and speaker recognition. Then, we give a more formal and detailed explanation of transformer networks [24], Wav2Vec2 [5], and CTC [12].

Automatic speech recognition. The task of automatic speech recognition (ASR) can be described as automatic translation of spoken language into text. The growing amounts of available computing power in combination with the rise of deep neural networks made for groundbreaking successes in the ASR domain in recent years [5, 16, 20].

Neural networks were first used in ASR implementations in the late 1980's. However, these early neural networks were mainly able to classify short time-units such as phonemes and isolated words [25, 26]. This was solved by the introduction of recurrent neural networks, which allowed for translating longer input sequences. Current state-of-the-art networks such as Wav2Vec2 [5] often utilize transformer networks [24], which are able to capture contextual information over entire input sequences.

Speaker-change detection. We describe speaker-change detection (SCD) as the task of identifying the moments in an audio recording at which one speaker stops talking, and another speakers starts talking. Most approaches consider SCD to be a binary labeling problem, where all time steps of the input audio are labeled to either be a speaker change or not [1, 6]. Our approach to SCD is to translate the audio recording to text using ASR, and indicating the speaker changes in the transcribed text using a speaker-change token.

Speaker recognition. Speaker recognition describes the task of identifying a speaker from characteristics of voices. It covers both speaker identification (SI) and speaker verification (SV). SI is the task of determining an unknown speaker identity corresponding to an utterance. In practice, this is done by comparing the utterance or a representation of the utterance against multiple other utterances. Classic SI systems often apply a Gaussian mixture network, as described in [11, 22]. Modern approaches utilize the power of convolutional neural networks such as ResNet [13] and VGG [2].

Speaker verification describes the task of determining whether two utterances belong to the same speaker. These two utterances are referred to as the enrollment utterance, and the verification utterance. SV can be either text dependent, or text independent. Text dependent verification requires the speaker to speak the same text as the enrollment recording; text independent verification does not have any requirements on the enrollment recording or on the recording to be verified.

2.1 Transformer networks

Transformer networks have first been introduced by Vaswani et al. [24] and are now a well established state-of-the-art approach

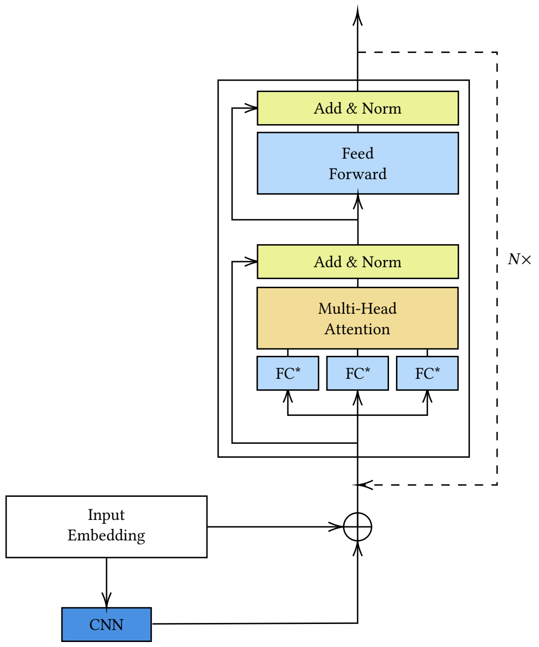

Figure 1: Graphical overview of a stack of encoder blocks as used in Wav2Vec2. In this figure the stack consists of N encoder blocks. Fully connected layers indicated by "FC*" have no bias term. This figure is an adaptation of the transformer network architecture by Vaswani et al. [24].

when it comes to sequence modeling. Using attention mechanisms, transformer networks are able to achieve superior performance compared to long short-term memory [14], and gated recurrent neural networks [8].

A transformer network consists of two main components: a stack of encoder blocks and a stack of decoder blocks. The main objective of the encoders is to transform input to a contextualized representation. The main objective of the decoders is to use the contextualized representation to generate output data. In this section, we will give an overview of the encoder block as used in the Wav2Vec2 network. We will not cover the decoder, as we do not use it in our experiments. For a detailed explanation of the attention mechanism, we refer to the work done by Vaswani et al. [24].

2.1.1 Transformer encoder. A single encoder block consists of two components: a multi-head attention mechanism, and a feed-forward network. A residual connection is employed around both components, which is followed by layer normalization [3]. The input embedding of the first encoder block in a stack of encoders is summed with a positional encoding. For Wav2Vec2, this positional encoding is constructed by forwarding the input embedding through a single layer CNN. Figure 1 gives a graphical overview of a stack of encoder blocks. In Section 2.2 we will go into further details on the positional encoding used in Wav2Vec2.

2.2 Wav2Vec2

In 2020, Baeveski et al. [4] showed the potential of pre-training on unlabeled data for speech processing by proposing the Wav2Vec2 network and its corresponding training procedure. In this section, we give a brief description of the Wav2Vec2 network architecture. In our research we only fine-tune pre-trained networks. Therefore, we will not cover its pre-training procedure. A graphical overview of the Wav2Vec2 network can be found in Figure 2.

Feature encoder. The feature encoder encodes raw audio waveforms into learned representation. It consists of seven blocks containing a temporal convolution followed by layer normalization. These blocks convert an audio recording $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$, to a sequence of feature vectors $\mathbf{Z} = \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$, each feature vector describing 20 milliseconds.

Positional encoding. Each feature vector in $\mathbf{Z}$ is independently normalized to zero mean and unit variance and projected using a fully connected layer. The resulting vectors are masked using SpecAugment [19]. The masked projected representations are convolved using a one-dimensional convolution with a kernel size of 128, a stride of 1, and padding of 64 resulting in the positional encoding. This encoding is summed with the original masked projected representation. The result is forwarded through the transformer of the network.

Transformer. The transformer processes its input through 12 transformer encoder blocks for the *base* version, or 24 transformer-encoder blocks for the *large* version of the Wav2Vec2 network. Each encoder block consists of a 12-headed multi-head attention mechanism and a two-layer feed forward network. The transformer produces a contextualized representation $\mathbf{C} = \mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_T$ capturing both local and global information on the entire input sequence $\mathbf{X}$. In our experiments, vectors from the contextualized representation are used as speaker embeddings.

Network head. In order to fine-tune a pre-trained Wav2Vec2 network, a randomly initialized fully-connected-layer is added on top of the transformer encoder. We will refer to this layer as the head of the network. Usually, this layer consists of n nodes, representing the tokens of a vocabulary. We define a vocabulary to be a set of size of n, consisting of tokens we allow our model to predict. In our experiments, the vocabulary consists of all lower-case case letters in addition to some special tokens such as a word delimiter token. We will come back the tokens we use when describing our fine-tuning pipeline in Section 3.2.

Forwarding embedding $\mathbf{c}_t \in \mathbf{C}$ through the network head yields output vector $\tilde{\mathbf{y}}_t$. A softmax is applied to this vector to interpret its values as probabilities, after which the argmax is taken to obtain the predicted token for time step t. Doing this for every embedding in $\mathbf{C}$ gives us a sequence of tokens $\hat{\mathbf{y}}$, also referred to as the predicted labeling.

2.3 Connectionist Temporal Classification

One problem that arises during the computation of the loss of an ASR system, is that we do not directly know how the words in the

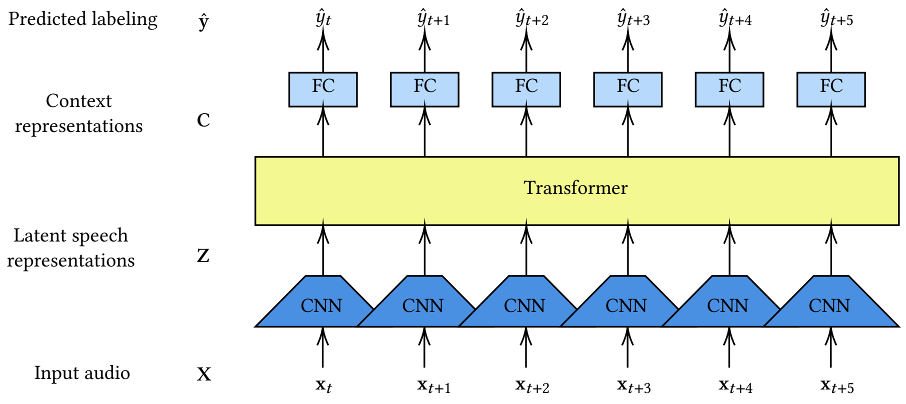
Predicted labeling $\hat{\mathbf{y}}$
Context representations $\mathbf{C}$
Latent speech representations $\mathbf{Z}$
Input audio $\mathbf{X}$
$\hat{y}_t$ $\hat{y}_{t+1}$ $\hat{y}_{t+2}$ $\hat{y}_{t+3}$ $\hat{y}_{t+4}$ $\hat{y}_{t+5}$
FC
Transformer
CNN
$\mathbf{x}_t$ $\mathbf{x}_{t+1}$ $\mathbf{x}_{t+2}$ $\mathbf{x}_{t+3}$ $\mathbf{x}_{t+4}$ $\mathbf{x}_{t+5}$

Figure 2: Overview of the Wav2Vec2 network architecture during fine-tuning. This figure is an adaptation of the Wav2Vec2 architecture by Baeveski et al. [5].

Pred. labeling	h	e	e	ϵ	l	ϵ	l	l	o
1. Collapse duplicates	h	e		ϵ	l	ϵ	l		o
2. Remove blanks	h	e			l		l		o
Pred. transcription	h e l l o								

Figure 3: Example of how the CTC mapping is used to convert a predicted labeling to a transcription.

ground truth transcription should be aligned with the labels predicted by a network. Connectionist Temporal Classification (CTC) [12] uses a many-to-one mapping accompanied by a loss function to solve alignment in sequence problems where timing is variable. The mapping can be used to map predicted labeling $\hat{\mathbf{y}}$ to a predicted transcription. The loss function is commonly used as a minimization objective when training ASR systems.

Many-to-one mapping. Before introducing the many-to-one mapping used to convert a predicted labeling to a predicted transcription, let us introduce some notations. Let V denote a vocabulary (i.e., set of tokens) and T the number of time steps. Now let V^T denote the set of sequences of length T consisting of tokens from vocabulary V. To allow for two or more equal consecutive tokens in the transcription, a blank token ϵ is introduced. Let V' denote the vocabulary including this blank token: $V' = V \cup \{\epsilon\}$. Using this notation, the many-to-one mapping CTC implements is defined as follows:

$$\mathcal{B} : V'^T \to V^{\leq T}, \tag{1}$$

where $V^{\leq T}$ denotes the set of sequences of length less than or equal to T. To map V'^T to $V^{\leq T}$, the following two steps are executed: first collapse all duplicate tokens, then remove all blank tokens. These two steps allow for different labelings to be mapped to the same transcription, making the mapping many-to-one. For an example of how the mapping is applied, we refer to Figure 3.

CTC-loss. The mapping described in the previous paragraph allows for a natural transition from probabilities at each time-step to the probability of a transcription text. For a given ground truth transcription $\mathbf{y} \in V^{\leq T}$ and input audio $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$, the conditional probability $p(\mathbf{y}|\mathbf{X})$ is given by the sum of the probabilities of all sequences corresponding to it:

$$p(\mathbf{y}|\mathbf{X}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{y})} p(\pi|\mathbf{X}), \tag{2}$$

where $\mathcal{B}^{-1}(\mathbf{y})$ computes all possible sequences corresponding to $\mathbf{y}$, and $p(\pi|\mathbf{X})$ is given by the product of posterior probabilities corresponding to sequence π. The CTC loss is computed by taking the negative logarithm over the result of Equation 2:

$$\mathcal{L}(\mathbf{y}, \mathbf{X}) = -\log p(\mathbf{y}|\mathbf{X}). \tag{3}$$

3 EXPERIMENTAL SETUP

The objective of this research is to fine-tune a Wav2Vec2-base model to perform three distinctive tasks: speech recognition, speaker-change detection, and speaker recognition. Our approach to this is to include speaker-change tokens and speaker identity tokens to the transcriptions during fine-tuning. To be able to apply speaker-change detection, we artificially construct multi-utterance datasets. In this section, we first explain how we construct these datasets and their corresponding transcriptions. Then, we introduce our training pipeline and explain how we evaluate models with respect to each of the three distinctive tasks.

3.1 Data

For our experiments, we use the LibriSpeech-clean-100 dataset [18]. We split the clean-100 set in a train, and validation set such that the two sets have no common speakers. The validation set is used to evaluate the performance during training. We use the clean development set from LibriSpeech to evaluate trained models during our research; and the clean test set from LibriSpeech to evaluate trained models after fixing all hyperparameters. We refer to these datasets as single-utterance datasets, as audio samples in these datasets consist of a single utterance.

3.1.1 Multi-utterance data. To conduct experiments related to speaker-change detection, we require a transcribed multi-utterance dataset. Since such datasets are hard to find, we artificially create multi-utterance datasets by combining samples from LibriSpeech-clean-100. The multi-utterance datasets are constructed by first selecting samples from a source dataset according to a selection criterion until the total duration of selected samples is at least 17.5 seconds. Then, the utterances and transcriptions of the selected samples are concatenated to obtain a multi-utterance sample. This is done until every sample of the source dataset is used exactly once. Note that this implies that there may be a multi-utterance sample that is less than 17.5 seconds long.

We consider three selection criteria, giving us three variants of multi-utterance datasets. The three selection criteria are as follows:

(1) **Single-participant:** A selected sample must be from the same speaker as the previously selected sample. This gives us single-participant samples consisting of multiple utterances. For this criterion, we consider two additional constraints, giving us two variants of single-participant multi-utterance datasets:

(a) **Same session:** A selected sample must follow the previously selected sample based on the LibriSpeech sample IDs. This gives us multi-utterance samples in which two adjacent utterances are recorded in the same recording session, have similar context, and are spoken by the same speaker. The resulting dataset is used during training to prevent the model learning to predict a speaker-change at the point where two recordings are concatenated.

(b) **Different session:** The selected sample must be from a different recording session than the previously selected sample. This gives us multi-utterance samples in which two adjacent utterances are from the same speaker but different recording sessions. The resulting dataset is used during evaluation to test to what extent our systems learn the actual speaker identities, and to what extent our systems learn to identify speakers based on other properties of the recordings, such as background noise or microphone used during recording.

Note that some speakers in the development and test set only have one recording session. Recordings of these speakers are not used in both variants of the single-participant dataset, as including them would not allow fulfilling the criterion that two adjacent utterances must be from different recording sessions.

(2) **Multi-participant:** A selected sample must be from a different speaker as that of the previously selected sample. This results in multi-participant audio samples, where no two adjacent utterances are from the same speaker.

A summary of all datasets we consider for our experiments, with their total number of samples and average number of utterances per sample, is given in Table 1.

dataset	variant	samples	avg. utt.
train	single-utterance	22909	1.00
	single-participant (1a)	11039	2.08
	single-participant (1b)	10986	2.09
	multi-participant (2)	10968	2.09
validation	single-utterance	5630	1.00
	single-participant (1a)	2724	2.07
	single-participant (1b)	2711	2.08
	multi-participant (2)	2724	2.07
development	single-utterance	2703	1.00
	single-participant (1a)	698	3.13
	single-participant (1b)	873	2.50
	multi-participant (2)	860	3.14
test	single-utterance	2620	1.00
	single-participant (1a)	654	3.10
	single-participant (1b)	906	2.24
	multi-participant (2)	850	3.08

Table 1: Number of samples and average number of utterances per sample for each of the datasets we consider for our experiments.

type	transcription					
1		hello		world		goodbye
2	#	hello	#	world	#	goodbye
3	001	hello	002	world	001	goodbye

Table 2: Example of how the three different types of transcriptions are constructed. In this example, the multi-utterance sample consists of three concatenated utterances, each one word long. The first and third utterance belong to the same speaker with identity 001; the second utterance belongs to a speaker with identity 002.

3.1.2 *Transcription variants.*

For every sample in each of the datasets, we create three types of transcriptions based on the objectives of the multitask system. A small example of how the different types of transcriptions are constructed can be found in Table 2. We describe the three types of transcriptions as follows:

(1) In the first type, we do not identify speaker-changes or speaker identities in the transcription. Such a transcription is obtained by concatenating transcriptions corresponding to the samples in the multi-utterance sample. These transcriptions are mainly used to obtain a baseline model.

(2) The second type of transcriptions includes a special symbol before every sequence of transcriptions belonging to a single speaker. This symbol indicates the beginning of an utterance spoken by a different speaker as that of the preceding utterance. We refer to this symbol as the speaker-change token and use a hashtag symbol (#) to denote it. These transcriptions are used for our experiments related to speaker-change detection, and speaker recognition.

(3) In the third type of transcriptions, we replace the speaker-change tokens of the transcriptions of the second type with speaker identity tokens. Every speaker-change token is replaced by a identifier corresponding to the speaker of the transcription following the speaker-change token. These transcriptions are used for our experiments related to speaker recognition.

3.2 Fine-tuning

For our experiments, we fine-tune multiple Wav2Vec2-base networks. We follow a similar pipeline as described by the authors of Wav2Vec2 [5] to minimize the CTC-loss. We freeze the feature encoder of the model and train for $100,000$ steps using a tri-stage learning-rate-scheduler in combination with an Adam-optimizer [15]. The tri-stage learning rate scheduler linearly increases the learning rate in the first 10% of steps from 3×10^{-7} to 3×10^{-5}. For the next 40% of steps, the learning rate is kept at a constant value of 3×10^{-5}. In the last 50% of steps, the learning rate is exponentially decreased from 3×10^{-5} to 1.5×10^{-6}. When fine-tuning we use an effective batch-size of 8, meaning that we forward 8 samples and accumulate their gradients before performing backpropagation. During evaluation a batch-size of 1 is used.

We fine-tune models on a single-utterance dataset, and a multi-utterance dataset, where the multi-utterance dataset is constructed by unifying the single-participant same session dataset (criterion 1a) and the multi-participant dataset (criterion 2). Note that we do not train on single-participant different session data (criterion 1b). For both datasets we use during fine-tuning, we fine-tune models using one of the three types of transcription, each requiring a different sized vocabulary and model head:

(1) The head of models trained on transcriptions that do not include speaker-change tokens or speaker identity tokens (transcription type 1), consist of 32 nodes: one node for each letter in the English alphabet and one node for each of the six special tokens: padding, unknown, begin sentence, end sentence, word delimiter, and apostrophe token. In practice the unknown, begin sentence, and end sentence tokens are not used but are left in the vocabulary due to code reuse.
(2) The head of models trained on transcriptions including speaker-change tokens (transcription type 2), consist of 33 nodes: the same 32 nodes as we had in the previous case, with one additional node representing a speaker-change token.
(3) When fine-tuning a model on transcriptions including speaker identities (transcription type 3), a head of 235 nodes is used: the same 33 nodes as we had in the previous case, with an additional node for every speaker identity in the train set. The node corresponding to the speaker-change token is not directly used when fine-tuning on transcriptions of type 3.

3.3 Evaluation

The fine-tuned models are evaluated based on their performance on each of the three tasks we consider. For all tasks, evaluation is done by comparing the text output by the model (also referred to as the hypothesis, or predicted transcription) with a ground truth transcription (also referred to as the reference).

We found that, in case the model is fine-tuned to perform speaker-change detection, it often places multiple adjacent speaker-change tokens in the hypothesis (that is, after applying the CTC procedure). This motivated us to add a post-processing step, replacing any subsequent speaker-change tokens in the hypothesis by a single speaker-change token. After this step, the hypothesis is aligned with the reference using dynamic string alignment; allowing the two to be compared at word level.

3.3.1 Automatic-speech-recognition. To evaluate the performance of the models with respect to ASR, we use the commonly used word-error-rate (WER). Given the aligned reference and hypothesis, the WER is computed as follows:

$$\text{WER} = \frac{S + D + I}{|\text{reference}|}, \tag{4}$$

where S denotes the number of substitutions, D the number of the deletions, I the number of insertions, and $|\text{reference}|$ the number of words in the reference. The operations (substitutions, deletions, and insertions) refer to how to get from the reference to the hypothesis.

When computing the WER, speaker-change tokens and speaker identity tokens are removed from both the hypothesis and the reference. By doing this, incorrect predicted speaker-changes or speaker identities are not taken into account during computation of the WER.

3.3.2 Speaker-change detection. In order to evaluate whether our models are able to accurately detect speaker-changes, we introduce the false positive rate and the false negative rate. For our problem domain, we define a false negative to be a speaker-change which has not been found by the model. After aligning the reference and the hypothesis, the number of false negatives is equal to the number of indices at which there is a speaker-change in the reference but not in the hypothesis. Given the false negatives, we compute the false negative rate (FNR) by dividing the number of false negatives by the total number of speaker-changes in the reference:

$$\text{FNR} = \frac{\text{nr. false negatives}}{\text{nr. speaker-changes}} \tag{5}$$

We define a false positive to be an incorrectly predicted speaker-change. I.e., after aligning the reference and hypothesis, the number of false positives is equal to the number of indices at which the model predicted a speaker-change which is not in the reference. The false positive rate (FPR) can be computed by dividing the total number of false positives by the number of words in the reference that are not a speaker-change token:

$$\text{FPR} = \frac{\text{nr. false positives}}{|\text{reference}| - \text{nr. speaker-changes}}. \tag{6}$$

3.3.3 Speaker recognition. The third objective of our models is speaker recognition. To evaluate how well our systems are able to recognize speaker identities, we use the equal-error-rate (EER). Computing the EER requires us to compute an embedding for every speaker-change token in the predicted transcriptions. Since we train our models to produce text, computing speaker embeddings is not a trivial procedure. We identify two challenges when computing embeddings. First, recordings may consist of multiple utterances from potentially different speakers, meaning that more than one embedding may need to be computed for a single recording. Second,

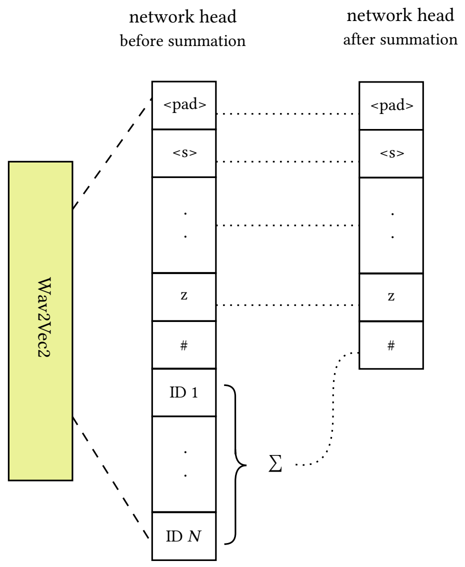

Figure 4: Graphical overview of how the model head is altered during evaluation in case the model is trained using transcriptions including speaker identities. In this example, the model is trained on a dataset consisting of N speaker identities. During evaluation, the corresponding posterior probabilities are summed to obtain the posterior probability of a speaker-change.

a sequence of speaker-change tokens is reduced to a single speaker-change token by the CTC procedure. For such sequences, we need to compute a single embedding.

Our approach to obtaining embeddings is to extract embeddings from the model at time steps where the posterior probability of a speaker-change is maximal. This implies that models trained on transcriptions of type 3, should produce transcriptions of type 2. We are able to do this by altering the head of the model during evaluation. In the remainder of this section, we first show how we alter the model head. Then, we explain how we extract embeddings from models producing transcriptions including speaker-changes. Finally, we explain how the extracted embeddings can be used to compute the EER.

Altering the head. Models trained on transcriptions including speaker identity tokens (transcription type 3) have a head consisting of 235 nodes, of which 202 correspond to speaker identities. During evaluation, nodes corresponding to speaker identities cannot be directly related to the speaker identities in the evaluation-set. Instead, we use these nodes by summing their corresponding posterior probabilities and interpreting the result as the posterior probability of a speaker-change. By doing this, models trained on transcriptions of type 3, produce transcriptions of type 2 during evaluation. A graphical overview of how the model head is altered during evaluation can be found in Figure 4.

Obtaining embeddings. To obtain embeddings for a given recording $\mathbf{X}$, we first forward the recording through the network, giving us predicted labeling $\hat{\mathbf{y}} = \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T$. Let $P(\hat{y}_i|\mathbf{X})$ denote the posterior probability of token $\hat{y}_i$ given recording $\mathbf{X}$. Furthermore, let $\hat{\mathbf{y}}' = \hat{y}_m, \hat{y}_{m+1}, \ldots, \hat{y}_n$ denote a subsequence of $\hat{\mathbf{y}}$ with a minimum length of 1, consisting only of speaker-change tokens. Note that such a sequence is reduced to a single speaker-change token by the CTC procedure. Given a sequence of speaker-change tokens, the time step at which the posterior probability is maximal is used to extract a speaker embedding $\mathbf{e}$:

$$\mathbf{e} = \mathbf{c}_t, \tag{7}$$

$$\text{where } t = \operatorname{argmax}\left(P(\hat{y}'_m|\mathbf{X}), P(\hat{y}'_{m+1}|\mathbf{X}), \ldots, P(\hat{y}'_n|\mathbf{X})\right) + m, \tag{8}$$

and $\mathbf{c}_t$ denotes the vector output by the final encoder block of the Wav2Vec2 model at time step t. Doing this for every mutual exclusive sequence of speaker-change tokens in $\hat{\mathbf{y}}$ gives an embedding for every speaker-change token in the hypothesis. The intuition behind this method is that time steps at which the posterior probability of a speaker change is maximal, are most representative for the speaker-change tokens in the hypothesis.

To be able to deal with false positive and false negative speaker-change predictions, we only use the embeddings in the computation of the EER, if the number of speaker-change tokens in the hypothesis is equal to the number of speaker-change tokens in the reference. Note that this implies that we may not be able to use all embeddings we find.

Computing the equal-error-rate. To be able to compute the EER we use a trial list. Such a list consists of pairs of utterances with a label indicating whether the two utterances are spoken by the same speaker. We compute a score for every pair in the list based on how similar the embeddings are that we extracted for the corresponding utterances. We use the cosine similarity to compute this score. After computing a score for every pair, a threshold z is set. Based on this threshold and the computed scores, pairs are labeled to be spoken by the same speaker or by different speakers. The resulting labeling is compared against the ground truth labeling. For the EER, z is chosen such that the false acceptance rate and false rejection rate are equal. Having a predicted labeling and ground truth labeling, the EER is computed as the fraction of incorrectly labeled pairs.

When computing the EER, we only use the trials for which both utterances in the pair an embedding is found. In case of false positive or false negative speaker-change predictions, we are not able to compute an embedding for every utterance. In such cases, a subset of the full trial list is used, which is why we are not able to directly compare the EER between different models and evaluation sets.

4 RESULTS

4.1 Baseline

To obtain baseline results, we fine-tune two Wav2Vec2-base networks: one on the single-utterance dataset and one on the multi-utterance dataset (union of single-participant same session data and multi-participant data). During fine-tuning, we use transcriptions that do not include speaker change symbols or speaker identities (transcription type 1). Performances of the fine-tuned models on the various development, and test sets can be found in Table 3.

The model fine-tuned on single-utterance samples, obtains a WER of 5.83% on both the single-utterance development set and test set.

train-set	evaluation-set	WER(%)	
		dev	test
single-utt.	single-utterance	5.83	5.83
	single-participant (1a)	5.85	6.12
	single-participant (1b)	6.04	6.31
	multi-participant (2)	7.78	7.75
multi-utt.	single-utterance	5.66	5.84
	single-participant (1a)	5.33	5.84
	single-participant (1b)	6.30	6.78
	multi-participant (2)	8.41	8.91

Table 3: WER on the test, and development sets for models fine-tuned on multi-utterance samples and single-utterance samples. Transcriptions that are used during fine-tuning and evaluation do not contain speaker change symbols or speaker identities.

Similar results are obtained on the single-participant same session datasets (criterion 1a). We observe a slight increase in WER when evaluating the model on single-participant different session data (criterion 1b); and a significantly higher WER when evaluating the model on the multi-participant samples.

For the model fine-tuned on multi-utterance samples, we observe the same trend. This shows that the task of translating audio into text is harder in case the recording consists of multiple utterances spoken by different speakers. It hints towards the model using speaker-related attributes to translate audio into text.

4.2 Speaker-change recognition

Our approach to speaker recognition requires the model to detect speaker-changes. In this section, we aim to test to what extent our models are able to detect speaker-changes. We fine-tune two Wav2Vec2-base networks using transcriptions including speaker-change tokens (transcription type 2). The word error rates, false positive rates, and false negative rates of the resulting models on the various development and test sets are denoted in Table 4.

First, let us consider the performance of the model fine-tuned on single-utterance samples. For all evaluation sets, this model obtains a similar WER as the baseline single-utterance model. This is expected, as for reference transcriptions of single utterance samples there is only one speaker-change token, which is placed at the beginning of the reference. The model is able to learn to always place this token at the start of the transcription without influencing the remainder of its prediction. The same reasoning can be applied when discussing the high FNR on the evaluation sets with no repeating speakers. The model always places a speaker-change token at the start, but never learned to place it at moments the speaker changes in the recording.

Next, let us discuss the results obtained by the model fine-tuned on the multi-utterance dataset. Comparing the WER of this model with that of the multi-utterance baseline model, we make three observations: firstly, the WER on single-utterance datasets is similar. Secondly, we see a slight improvement in WER when evaluating the model on single-participant data, where adjacent utterances are taken from different sessions (criterion 1b). This improvement is not seen in the other single-participant dataset (criterion 2). A final observation we make is that the model fine-tuned using transcriptions including speaker-change symbols obtains a lower WER on the multi-participant dataset than the baseline model. I.e., by introducing speaker change symbols in the transcriptions during fine-tuning, we maintain a similar WER on single-utterance and single-participant samples while we improve the WER on multi-participant samples. This shows that in our setup, it is beneficial for the quality of the ASR system to include speaker-change tokens in the target transcriptions during fine-tuning.

Regarding speaker-change detection, for both models we observe an FNR of 0.00% over all single-participant and single-utterance datasets. An FNR of 10.06% and 11.67% is observed on the multi-participant development en test set respectively. Comparing this with the FPR we observe that the model is more likely to miss a speaker-change than to predict a speaker-change where no speaker-change is present in the recording. Further inspecting the FPR, we observe the highest FPR on the single-participant dataset, with adjacent utterances being from different sessions (criterion 1b). We argue that the model has some tendency to predict speaker changes due to change in context, background-noise, or use of different microphones during recording. However, we consider an FPR of 0.81% still to be relatively low. Additionally, the FPR on all other evaluation sets is approximately 0.00%.

4.3 Speaker recognition

In this section, we will describe and discuss the experiments related to speaker recognition. We fine-tune two Wav2Vec2-base networks using transcriptions including speaker-identity tokens (transcription type 3). As we did for our previous experiments, we fine-tune one model on single-utterance samples, and one model on multi-utterance samples. In Table 5 we give an overview of the performance both models obtain.

When computing the EER we use a subset of the full trial list. Computing a score of a given trial requires an embedding for both utterances in the trial. Due to false negatives and false positives, we do not have an embedding for all utterances. Therefore, we are not able to compute a score for all trials. Only the trials are considered where we found an embedding for both speaker identities. For the model fine-tuned on single-utterance samples, we only note the EER on the single-utterance dataset, as we do not expect this model to extract representative embeddings from multi-utterance samples. Since different subsets of the trial list are used for the different evaluation sets, we are not able to compare the EER between different models and evaluation sets.

Let us first discuss the results of the model fine-tuned on single utterance-samples. For this model, we observe an EER of 10.58% for the single-utterance development-set, and 7.23% for the single-utterance test set. Since both rates are well below 50%, we argue that fine-tuning on transcriptions including speaker identities allows extracting embeddings that capture identities of unknown speakers.

Comparing the model fine-tuned on single-utterance samples with the baseline model, we observe a slight increase in WER. This is not unexpected, as we make the fine-tuning process more challenging by introducing speaker-identity tokens in the target transcriptions,

train-set	evaluation-set	WER (%)		Δ WER (%)		FPR(%)		FNR(%)	
		dev	test	dev	test	dev	test	dev	test
single-utterance	single-utterance	5.70	5.91	−0.14	0.08	0.00	0.00	0.00	0.00
	single-participant (1a)	5.74	6.01	−0.11	−0.11	0.00	0.00	0.00	0.00
	single-participant (1b)	5.87	6.29	−0.17	−0.02	0.00	0.00	0.00	0.00
	multi-participant (2)	7.68	7.70	−0.10	−0.05	0.00	0.00	68.18	67.56
multi-utterance	single-utterance	5.69	5.93	0.01	0.09	0.00	0.00	0.00	0.00
	single-participant (1a)	5.61	5.72	−0.69	−0.96	0.05	0.07	0.00	0.00
	single-participant (1b)	5.97	5.93	−0.33	−0.85	0.81	0.68	0.00	0.00
	multi-participant (2)	5.97	6.18	−2.44	−2.73	0.00	0.02	10.06	11.67

Table 4: WER, FPR, and FNR on the test, and development set for models fine-tuned on multi-utterance samples and regular LibriSpeech-clean-100 samples. Transcriptions that are used during fine-tuning and evaluation contain speaker change symbols; when computing the WER, these are removed from both the reference and the hypothesis. The Δ WER column shows the difference with the corresponding baseline models.

train-set	evaluation-set	WER (%)		Δ WER (%)		FPR (%)		FNR (%)		EER (%)*	
		dev	test	dev	test	dev	test	dev	test	dev	test
single-utterance	single-utterance	5.83	6.11	−0.01	0.28	0.00	0.00	0.55	0.65	10.58	7.23
	single-participant (1a)	5.88	6.19	0.03	0.07	0.00	0.00	1.00	0.76	-	-
	single-participant (1b)	6.01	6.40	−0.03	0.09	0.00	0.00	1.72	0.44	-	-
	multi-participant (2)	7.95	7.95	0.17	0.20	0.00	0.00	68.44	67.98	-	-
multi-utterance	single-utterance	5.63	6.07	−0.05	0.09	0.00	0.01	0.22	0.31	10.90	8.67
	single-participant (1a)	5.55	5.58	0.22	−0.26	0.03	0.03	0.28	0.00	11.26	6.59
	single-participant (1b)	5.94	6.37	−0.36	−0.41	0.69	0.56	0.33	0.53	4.15	5.97
	multi-participant (2)	6.10	6.35	−2.31	−2.56	0.00	0.02	7.88	9.95	14.23	9.48

Table 5: WER, FPR, FNR, and EER on the test, and development set for models fine-tuned on multi-utterance samples and regular LibriSpeech-clean-100 samples. Transcriptions that are used during fine-tuning contain speaker identities; when computing the WER, these are removed from both the reference and the hypothesis. *EER is computed using a subset of the trials; therefore, the EER should not be compared between models and evaluation sets.

but use the same amount of fine-tuning steps as we did when fine-tuning the baseline model. We use the same argumentation for explaining the increase in FNR on the single-utterance, single-participant data. The high FNR on the no-repeating-samples can be explained following the same argumentation we used when discussing the results obtained by the model fine-tuned using transcriptions including speaker changes (transcription type 2): the model never learned to predict speaker-change token other than the speaker-change token placed at the start of the target transcription.

Next, let us discuss the model fine-tuned on multi-utterance samples and compare it against the results of our previous experiments. Again, we see that the model is able to obtain equal-error-rates well below 50%, meaning that we are able to extract embeddings capturing speaker identities. For the WER we see a similar phenomenon as we saw for the model fine-tuned using transcriptions of type 2: by introducing speaker identities in the transcriptions, we lower the WER on multi-participant samples. This verifies that in our setup, it is beneficial for the quality of speech-recognition to include speaker-change symbols or speaker identities in the transcriptions.

4.3.1 Assumptions. When fine-tuning on samples including speaker identities, we made two assumptions. First, embeddings extracted at time steps at which a speaker changes are predicted capture speaker identities better than embeddings at different time steps. Second, embeddings outputted by the final encoder block of the Wav2Vec2-base network capture identities better than embeddings outputted by other encoder blocks. While experiments showed that the embeddings we extracted capture speaker identities, we have not yet considered using other time steps or other encoder blocks to extract embeddings. In section A of the appendix, we describe two sets of experiments regarding both assumptions we made.

5 CONCLUSION

This work showed that the Wav2Vec2 network can successfully be fine-tuned on an artificially created multi-utterance dataset to perform three distinctive tasks: speech-recognition, speaker-change-detection, and speaker-recognition. Our approach to this multitask system relied on the introduction of speaker-change tokens or speaker-identity tokens in the transcriptions during fine-tuning. The introduction of these tokens allowed us to obtain a lower WER on multi-participant recordings while maintaining similar WER on single-participant and single-utterance samples. Furthermore, our approach allowed us to extract speaker embeddings that can effectively be used for speaker-recognition.

The lack of transcribed multi-utterance audio datasets required us to create a multi-utterance audio dataset artificially. We must note that training and evaluating models using our created datasets introduces a number of downsides. Most notable is that our trained models do not generalize well to other datasets such as VoxCeleb [17]. This causes our results not to be comparable to research in the field of speaker-recognition or speech-recognition. Future research can tackle these downsides by transcribing a multi-participant or a speaker-diarization dataset and applying our approach on this.

A ADDITIONAL EXPERIMENTS: SPEAKER RECOGNITION

For our experiments regarding speaker recognition (Section 4.3) we made two assumptions:

(1) Embeddings extracted at time steps at which speaker changes are predicted capture speaker identity.
(2) Embeddings extracted from the last encoder block of the Wav2Vec2-Base network capture speaker identities better than embeddings extracted from other encoder blocks.

In this section, we test these assumptions by conducting two sets of experiments.

Assumption 1. For the results discussed in Section 4.3 we trained models on transcriptions including speaker identities. Since, we do not know the identities of speakers in the evaluation sets, we sum the posterior probabilities corresponding to speaker identities, and interpret the result as the posterior probability of a speaker change. We saw that during training, the model is able to learn the speaker identities in the train set as it is able to put the correct speaker identity in the transcription. During evaluation, we made the assumption that embeddings at time steps where the sum of posteriors corresponding to training identities is larger than all other posteriors, capture the identities of speakers in the evaluation set. In this section, we test whether this assumption holds by computing the EER for three different sets of embeddings. The first set of embeddings is extracted following the pipeline described in Section 3.3.3. The second set of embeddings is constructed by selecting an embedding from a random time step for every sample. The third set of embeddings is constructed by averaging the embeddings found at each time step for every sample.

All sets are extracted from the model trained on single-utterance samples using transcriptions including speaker-identities. To allow for a more fair comparison between equal-error-rates, the same set of trials is used for all computations done on an evaluation set. The fraction of the trials that are not used is smaller than 1% for both sets. The results of the experiments are summarized in Table 6.

When using random time steps to extract embeddings, we obtain an EER of approximately 50% for both datasets. This makes the performance of this extraction method similar to random guessing. We see a significant decrease in EER when comparing this with the results obtained by averaging the embeddings over all time steps.

extraction method	**EER (%)***	
	dev	**test**
random time step	49.23	49.37
average over all time steps	29.37	30.06
pipeline Section 3.3.3	10.58	7.23

Table 6: Equal-error-rates of embeddings extracted in three different manners. *Due to some false-negative speaker-change predictions, a subset of the full trial list is used to compute the EER.

The lowest EER is obtained by extracting embeddings following the pipeline described in Section 3.3.3. This is not unexpected, as the model is trained to predict the speaker identity at the timestamp corresponding to a speaker-change.

Assumption 2. To test whether our second assumption holds, we extract embeddings from the outputs of each of the 12 transformer encoder blocks of the Wav2Vec2-Base network, as well as the input of the first encoder block. We use the model trained on single-utterance samples with transcriptions of type three to extract embeddings from single-utterance evaluation sets. Doing this for the outputs of each of the 12 encoder blocks, as well as the input of the first encoder block gives us 13 sets of embeddings for both the single-utterance dev-set, and the single-utterance test-set. For each set of embeddings, we compute the EER. Results of this experiment are plotted in Figure 5.

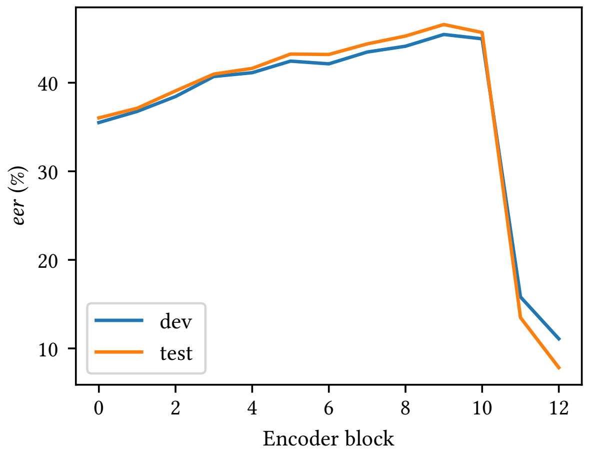

Figure 5: EER against the encoder block from which the embeddings are extracted. The encoder blocks are labeled based on their dept in the network: 0 indicates the input of the first encoder block, 1-12 the outputs of the first to last encoder blocks.

A first observation we make is that embeddings extracted from the last two encoder blocks of the network result in a significantly lower EER than embeddings extracted from encoder blocks earlier in the network. This shows that the assumption we made holds. A second observation we make is that the EER increases as we use embeddings from deeper encoder blocks (up till encoder block 10).

REFERENCES

[1] Ajmera, J., McCowan, I., and Bourlard, H. Robust speaker change detection. *IEEE Signal Processing Letters 11*, 8 (2004), 649–651.

[2] An, N. N., Thanh, N. Q., and Liu, Y. Deep cnns with self-attention for speaker identification. *IEEE Access 7* (2019), 85327–85337.

[3] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization.

[4] Baevski, A., Zhou, H., Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations.

[5] Baevski, A., Zhou, H., Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR abs/2006.11477* (2020).

[6] Bredin, H., Barras, C., et al. Speaker change detection in broadcast tv using bidirectional long short-term memory networks. In *Interspeech 2017* (2017), ISCA.

[7] Chen, X.-W., and Lin, X. Big data deep learning: Challenges and perspectives. *IEEE Access 2* (2014), 514–525.

[8] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* (2014).

[9] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* (2018).

[10] Fan, Z., Li, M., Zhou, S., and Xu, B. Exploring wav2vec 2.0 on speaker verification and language identification. *arXiv* (2020).

[11] Gish, H., and Schmidt, M. Text-independent speaker identification. *IEEE signal processing magazine 11*, 4 (1994), 18–32.

[12] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning* (2006), pp. 369–376.

[13] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[14] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[15] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv* (2014).

[16] Li, J., et al. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing 11*, 1 (2022).

[17] Nagrani, A., Chung, J. S., Xie, W., and Zisserman, A. Voxceleb: Large-scale speaker verification in the wild. *Computer Science and Language* (2019).

[18] Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. Librispeech: An asr corpus based on public domain audio books. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015), IEEE, pp. 5206–5210.

[19] Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv* (2019).

[20] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision. *arXiv* (2022).

[21] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training.

[22] Reynolds, D. A., and Rose, R. C. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on speech and audio processing 3*, 1 (1995), 72–83.

[23] Vaessen, N., and van Leeuwen, D. A. Fine-tuning wav2vec2 for speaker recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2022), pp. 7967–7971.

[24] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[25] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing 37*, 3 (1989), 328–339.

[26] Wu, J., and Chan, C. Isolated word recognition by neural network models with cross-correlation coefficients for speech dynamics. *IEEE transactions on pattern analysis and machine intelligence 15*, 11 (1993), 1174–1185.

[27] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems 32* (2019).